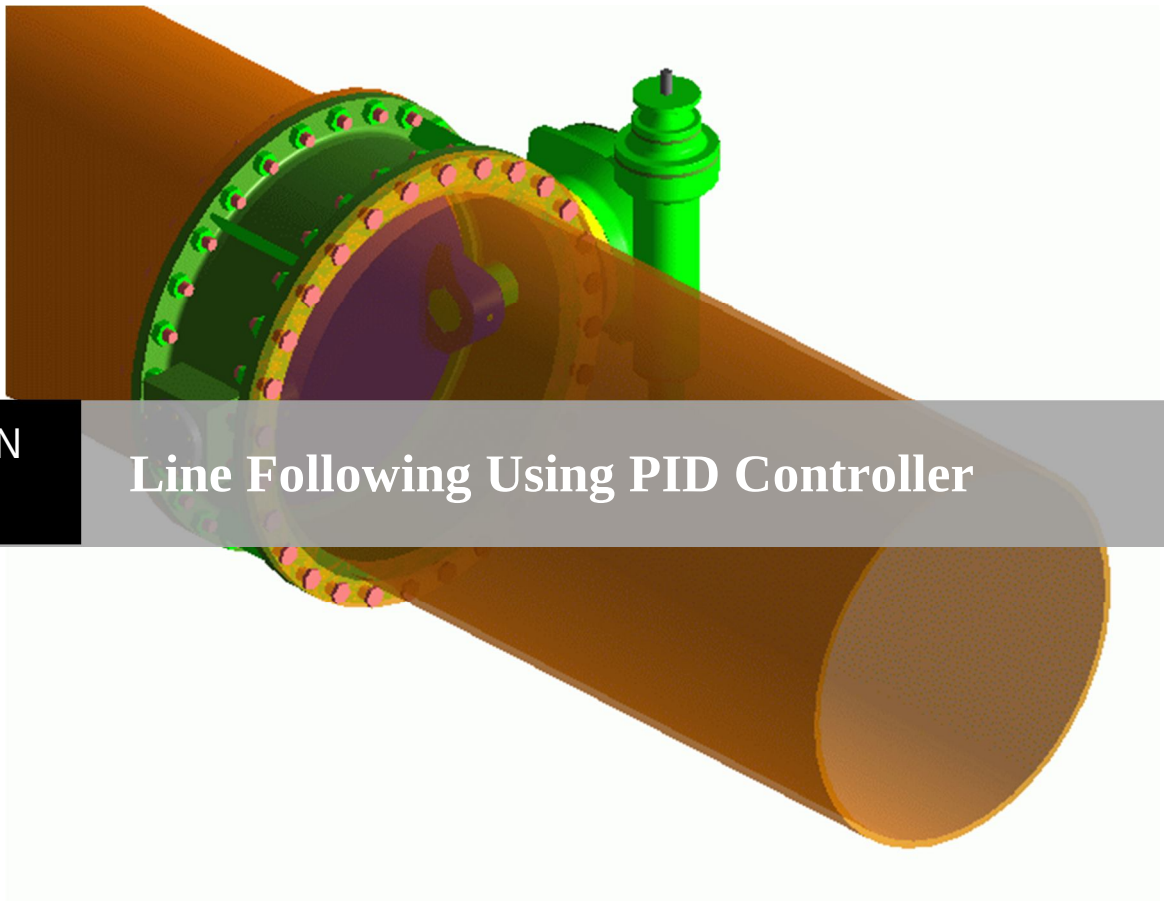
1/1/2012



ROBOCON 2012

Line Following Using PID Controller

| Soren Goyal and Vikas Kumar Singh

Acknowledgement

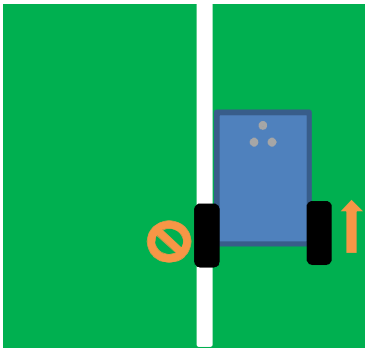
- Team Robocon IIT Kanpur
- Robotics Club
- Pratap Bhanu
- Raja C

Contact Us

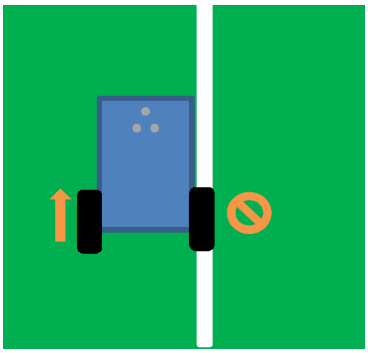
- soren@iitk.ac.in
- vksingh@iitk.ac.in

Consider **Line Following** using two motors.
Simplest line following algorithm simply looks to see if the robot is to the right or left of the line.

1. If the robot is to the right of the line, the left motor stops and the right motor speeds up.



2. If the robot is to the left of the line, the left motor speeds up and the right motor stops.

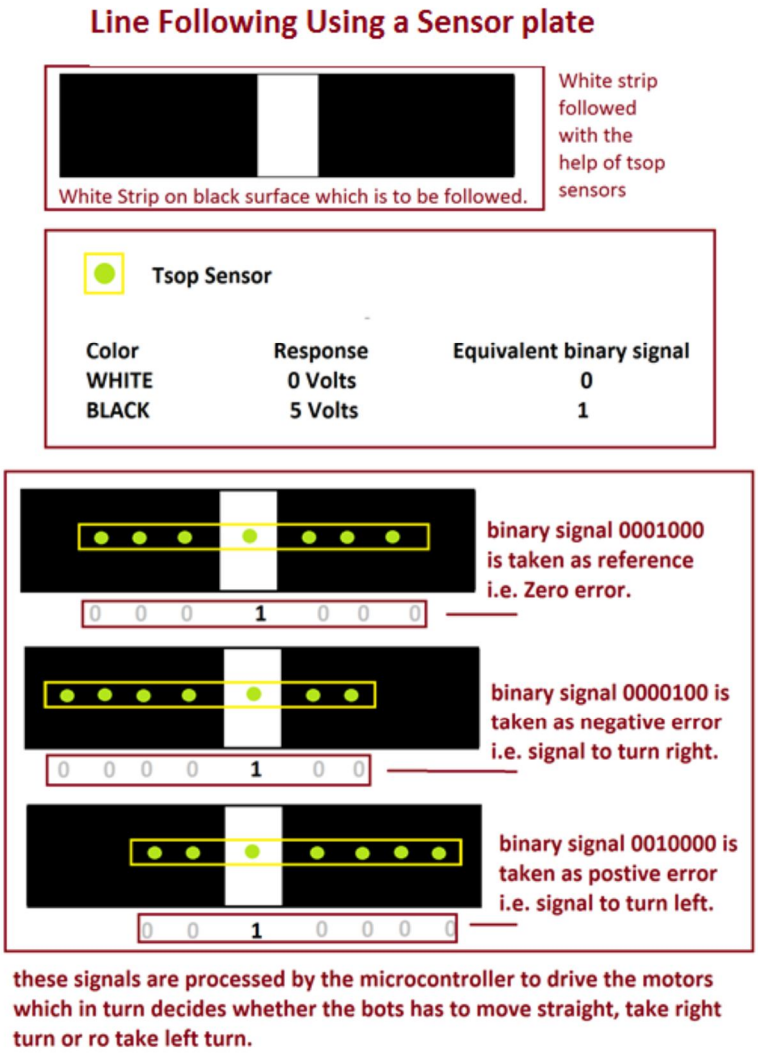


This gets the job done. But the robot tends to move in a 'zig-zag' path. The maximum speed achieved by this algorithm is less compared to what can be achieved by a PID based algorithm. Also line following on turns is unreliable unless the robot is slowed down.

A better implementation of this algorithm would be to calculate the deviation of the robot from the line. Then increase or decrease the motor speeds proportional to the deviation

e.g-the deviation ranges between -3 to +3.

soren@iitk.ac.in



RPM of Left motor=MEAN RPM –(k_p*deviation)

RPM of Right motor=MEAN RPM +(k_p*deviation)

Here 'k_p' is the constant of proportionality. It has to set experimentally. So, if the robot is to the left of the line, the deviation will be negative. Consequently, the RPM of the left motor will increase and the right motor will decrease.

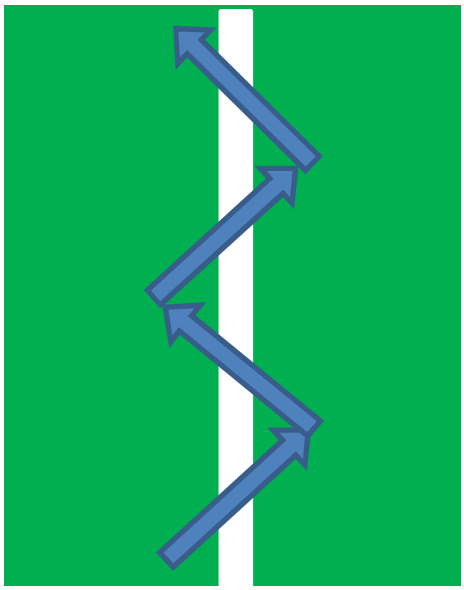
This algorithm gives stable line following, and for most cases this would be sufficient. So, unless you want to turn on high speeds while clinging tightly to the line this would work for you.

soren@iitk.ac.in

This has 2 **problems**-

1. If the robot is not responsive enough it will return back slowly to the mean position even after deviation is detected .

2. Now, you may want to increase the constant of proportionality, so that the return to mean position is faster. But this would mean that when the robot reaches the mean position it will tend to overshoot it due to inertia-cross to the other side-return back with a high speed-then over shoot again. Resulting in a zig–zag path.



Zig-zag line following. Stable but slow and rough

THE PID CONTROLLER ALGORITHM

A **proportional–integral–derivative controller** (**PID controller**) is a generic feedback controller. It can be used whenever a mean position has to be achieved but the controls of the system do not react instantaneously and accurately.

The PID algorithm, takes in account the following 3 things- the existing error, the time the system has stayed away from the mean position and the possibility of overshooting the mean position. Using these 3 quantities, the system is controlled better, allowing it to reach the mean position faster and without over shooting it.

Two terms before we use PID for line following-

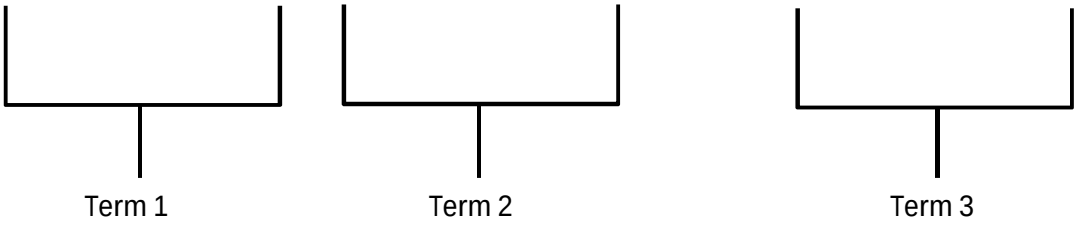
Correction - The term that is added and subtracted to the RPMs of the motors.

soren@iitk.ac.in

Deviation – The deviation of the robot from the line. It is negative when the robot is to the left of the line and positive when it's to the right.

PID is used to calculate the 'correction' term.

Correction = $k_p * deviation + k_i * \int deviation.\, dt + k_d * \frac{d}{dt}(deviation)$



K_p, k_i and k_d are constants which are set experimentally.

If only the first term had been used to calculate the correction, the robot would have reacted in the same way as in the classical line following algorithm.

The second term forces the robot to move towards the mean position faster. The third term resists sudden change in deviation.

The Code

The code for calculating deviation. Here 'a' is a byte which represents the state of the sensors.So, if 'a' in binary is '00000001' the line line is below the rightmost sensor. This code follows the opposite convention for sign of deviation, i.e if the robot is to the right of the line then the deviation is negative.

soren@iitk.ac.in

```
if(a==0b0100000) deviation=-6;
if(a==0b01100000)deviation=-5;
if(a==0b00100000)deviation=-4;
if(a==0b00110000)deviation=-3;
if(a==0b00010000)deviation=-2;
if(a==0b00011000)deviation=-1;
if(a==0b00001000)deviation=0;
if(a==0b00001100)deviation=1;
if(a==0b00000100)deviation=2;
if(a==0b00000110)deviation=3;
if(a==0b00000010)deviation=4;
if(a==0b00000011)deviation=5;
if(a==0b00000001)deviation=6;
```

The integral term is simply the summation of all previous deviations. Call this integral- 'totalerror'. The derivative is the difference between the current deviation and the previous deviation. Following is the code for evaluating the correction. These lines should run in each iteration

```
correction=kp*deviation + ki*totalerror + kd*(deviation-previousdeviation);
totalerror+=correction;
previousdeviation=deviation;
```


soren@iitk.ac.in